CS 242

Programming Languages

John Mitchell

Course web site: http://www.stanford.edu/class/cs242/

A little about myself …

Research Interests:

Computer security: access control, cryptographic protocols and mobile code security.

Programming languages, type systems, object systems, and formal methods.

Applications of logic to CS.

B.S. Stanford University; M.S., Ph.D. MIT.

◆ How I spend my time
- Teaching classes
- Working with graduate students
- Writing papers, going to conferences, giving talks
- Departmental committees, other stuff outside Stanford: conferences organization, journals, public service, …

Course Goals

◆ Programming Language Culture
- A language is a “conceptual universe” (Perlis)
  - Learn what is important about various languages
  - Understand the ideas and programming methods
- Understand the languages you use (C, C++, Java) by comparison with other languages
- Appreciate history, diversity of ideas in programming
- Be prepared for new programming methods, paradigms

◆ Critical thought
- Properties of language, not documentation

◆ Language *and* implementation
- Every convenience has its cost
  - Recognize the cost of presenting an abstract view of machine
  - Understand trade-offs in programming language design

Transference of Lang. Concepts

◆ Parable
- I started programming in 1970’s
  - Dominant language was Fortran; no recursive functions
- My algorithms and data structure instructor said:
  - Recursion is a good idea even though inefficient
  - You can use idea in Fortran by storing stack in array
- Today: recursive functions everywhere

◆ Moral
- World changes; useful to understand many ideas

◆ More current example: function passing
- Pass functions in C by building your own closures, as in STL “function objects”

Alternate Course Organizations

◆ Language-based organization
- Algol 60, Algol 68, Pascal
- Modula, Clu, Ada
- Additional languages grouped by paradigm
  - Lisp/Scheme/ML for functional languages
  - Prolog and Logic Programming
  - C++, Smalltalk and OOP
  - Concurrency via Ada rendez-vous

My opinion:
- Historical concepts are same across many languages
  - Block structure, scope, memory management
- OOP deserves greater emphasis

For comparison, see Sethi’s book …

Alternate Course II

◆ Concept-based organization
- Use single language like Lisp/Scheme
- Present PL concepts by showing how to define them

◆ Advantages:
- Uniform syntax, easy to compare features

◆ Disadvantages
- Miss a lot of the culture associated with languages
- Some features hard to add
  - Type systems, program-structuring mechanisms
  - Works best for “local” features, not global structure

Examples: Abelson/Sussman, Friedman et al.

Organization of this course

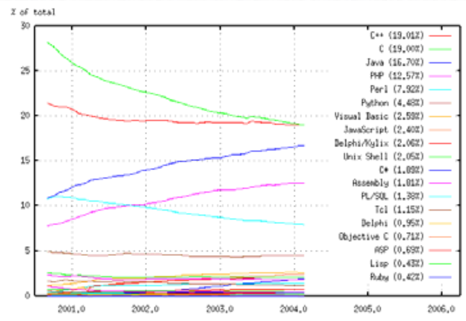
- Programming in the small
  - Cover traditional Algol, Pascal constructs in ML
    - Block structure, activation records
    - Types and type systems, ...
  - Lisp/Scheme concepts in ML too
    - higher-order functions and closures, tail recursion
    - exceptions, continuations
- Programming in the large
  - Modularity and program structure
  - Specific emphasis on OOP
    - Smalltalk vs C++ vs Java
    - Language design and implementation

Course Organization (cont'd)

- Concurrent and distributed programming
  - General issues in concurrent programming
  - Actor languages: an attempt at idealization
  - Concurrent ML
  - Java threads

Do we emphasize C?
  - Important, practical language
  - We discuss other languages, compare to C as we go
    - "Intro to C for Java programmers"?
  - We do cover the ++ part of C++ in some detail

Languages in common use (I)



Compiled by François Labelle from statistics on open-source projects at SourceForge

Languages in common use (II)

Position	Delta 1 Year	Programming Language	Ratings	Delta 1 Year
1	⬆	C	17.122%	-0.65%
2	⬇	Java	15.896%	-6.35%
3	=	C++	14.916%	-2.62%
4	⬆	(Visual) Basic	11.650%	+4.47%
5	⬇	Perl	8.968%	+0.87%
6	=	PHP	8.231%	+2.80%
7	⬆⬆⬆⬆	Delphi/Pascal/Kylix	5.865%	+4.28%
8	⬆⬆⬆⬆	Python	5.597%	+4.17%
9	⬇	SQL	2.693%	-0.55%
10	⬇	C#	1.634%	-0.43%

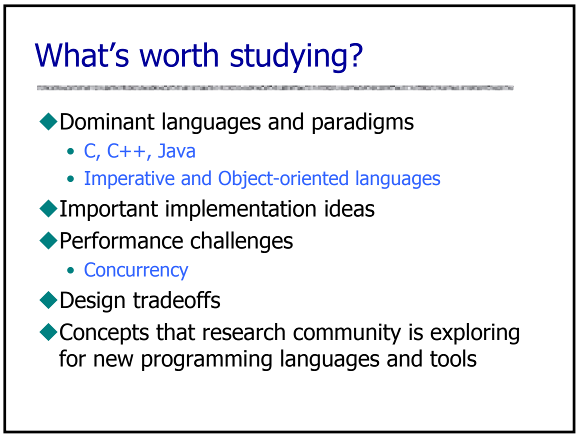
TPC index based on world-wide availability of skilled engineers, courses, and third party vendors, determined by using Google and Yahoo! search engines

Language groups

- Multi-purpose languages
  - C, C++, Java
  - Visual Basic
  - Object Pascal: Delphi, Kylix, …
  - Lisp, Scheme, ML
- Scripting languages
  - Perl, PHP
  - Shell
- Special-purpose languages
  - SQL
  - Prolog

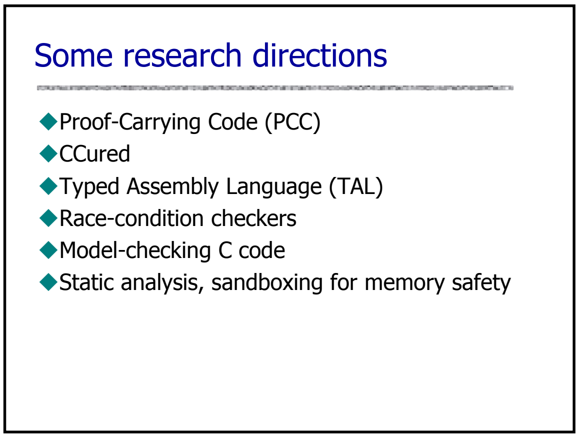
What's new in programming languages

- Commercial trend over past 5 years
  - Increasing use of Java, C#, … type-safe languages
  - Scripting languages, other languages for web applications
- Teaching trends
  - Java replacing C as most common intro language
    - Less emphasis on how data, control represented in machine
- Research and development trends
  - Modularity
    - Java, C++: standardization of new module features
  - Program analysis
    - Automated error detection, programming env, compilation
  - Isolation and security
    - Sandboxing, language-based security, …



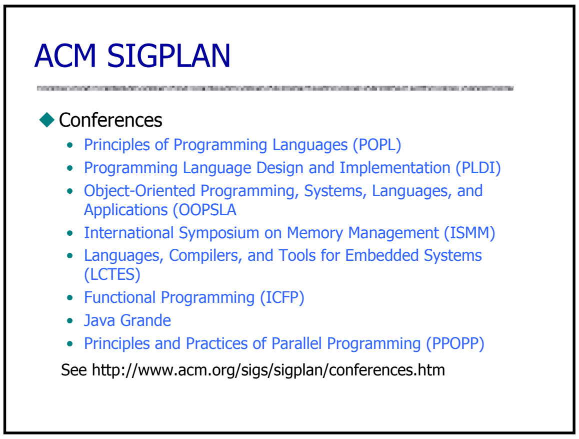
What's worth studying?

- Dominant languages and paradigms
  - C, C++, Java
  - Imperative and Object-oriented languages
- Important implementation ideas
- Performance challenges
  - Concurrency
- Design tradeoffs
- Concepts that research community is exploring for new programming languages and tools



Some research directions

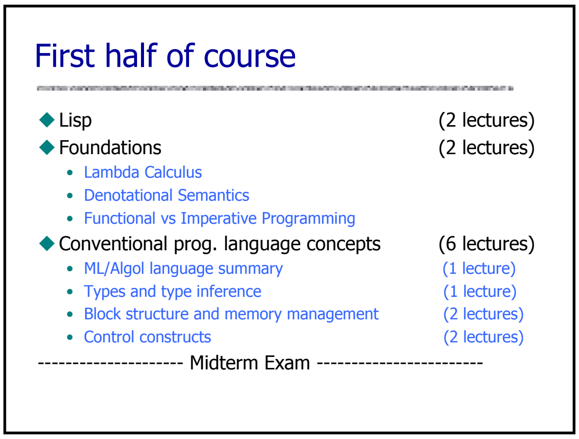
- Proof-Carrying Code (PCC)
- CCured
- Typed Assembly Language (TAL)
- Race-condition checkers
- Model-checking C code
- Static analysis, sandboxing for memory safety



ACM SIGPLAN

- Conferences
  - Principles of Programming Languages (POPL)
  - Programming Language Design and Implementation (PLDI)
  - Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA
  - International Symposium on Memory Management (ISMM)
  - Languages, Compilers, and Tools for Embedded Systems (LCTES)
  - Functional Programming (ICFP)
  - Java Grande
  - Principles and Practices of Parallel Programming (PPOPP)

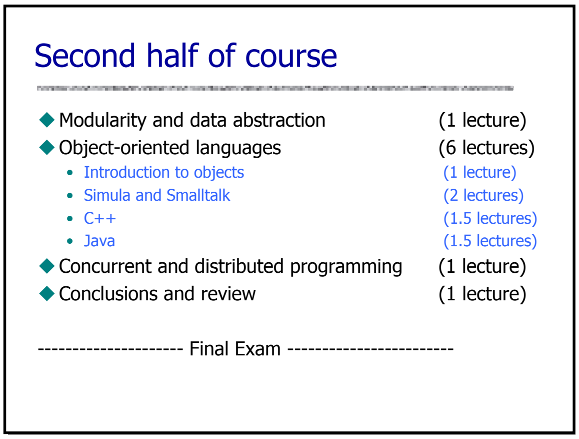
See http://www.acm.org/sigs/sigplan/conferences.htm



First half of course

Lisp	(2 lectures)
Foundations	(2 lectures)
• Lambda Calculus	
• Denotational Semantics	
• Functional vs Imperative Programming	
Conventional prog. language concepts	(6 lectures)
• ML/Algol language summary	(1 lecture)
• Types and type inference	(1 lecture)
• Block structure and memory management	(2 lectures)
• Control constructs	(2 lectures)

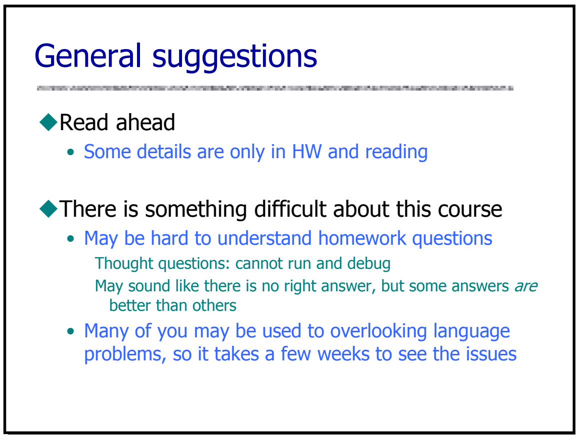
-------------------- Midterm Exam -----------------------



Second half of course

Modularity and data abstraction	(1 lecture)
Object-oriented languages	(6 lectures)
• Introduction to objects	(1 lecture)
• Simula and Smalltalk	(2 lectures)
• C++	(1.5 lectures)
• Java	(1.5 lectures)
Concurrent and distributed programming	(1 lecture)
Conclusions and review	(1 lecture)

-------------------- Final Exam -----------------------



General suggestions

- Read ahead
  - Some details are only in HW and reading
- There is something difficult about this course
  - May be hard to understand homework questions
    - Thought questions: cannot run and debug
    - May sound like there is no right answer, but some answers *are* better than others
  - Many of you may be used to overlooking language problems, so it takes a few weeks to see the issues

Course Logistics

- Homework and Exams
  - HW handed out and due on Wednesdays
  - Midterm Wed Oct 29 7-9PM ???, Final Monday Dec 8, 8:30AM
  - Honor Code, Collaboration Policy
- Homework grader?
  - Send email to cs242@cs email addr (operational shortly)
- TA's, Office hours, Email policy, …
- Section
  - Friday afternoons
  - Optional discussion and review; no new material
- Reading material
  - Book available in bookstore

Look at web site…

Questions?